MySQL

Enterprise applications

NetApp
May 03, 2024

Inhalt

MySQL

MySQL-Datenbanken auf ONTAP

MySQL und seine Varianten, darunter MariaDB und Percona MySQL, ist die weltweit beliebteste Datenbank.

Diese Dokumentation zu ONTAP und der MySQL-Datenbank ersetzt die zuvor veröffentlichte *TR-4722: MySQL-Datenbank unter ONTAP Best Practices.*

ONTAP ist eine ideale Plattform für MySQL-Datenbanken, da ONTAP buchstäblich für Datenbanken konzipiert ist. Es wurden speziell für die Anforderungen von Datenbank-Workloads zahlreiche Funktionen wie die Optimierung der zufälligen I/O-Latenz bis hin zur erweiterten Quality of Service (QoS) und grundlegenden FlexClone-Funktionalität erstellt.

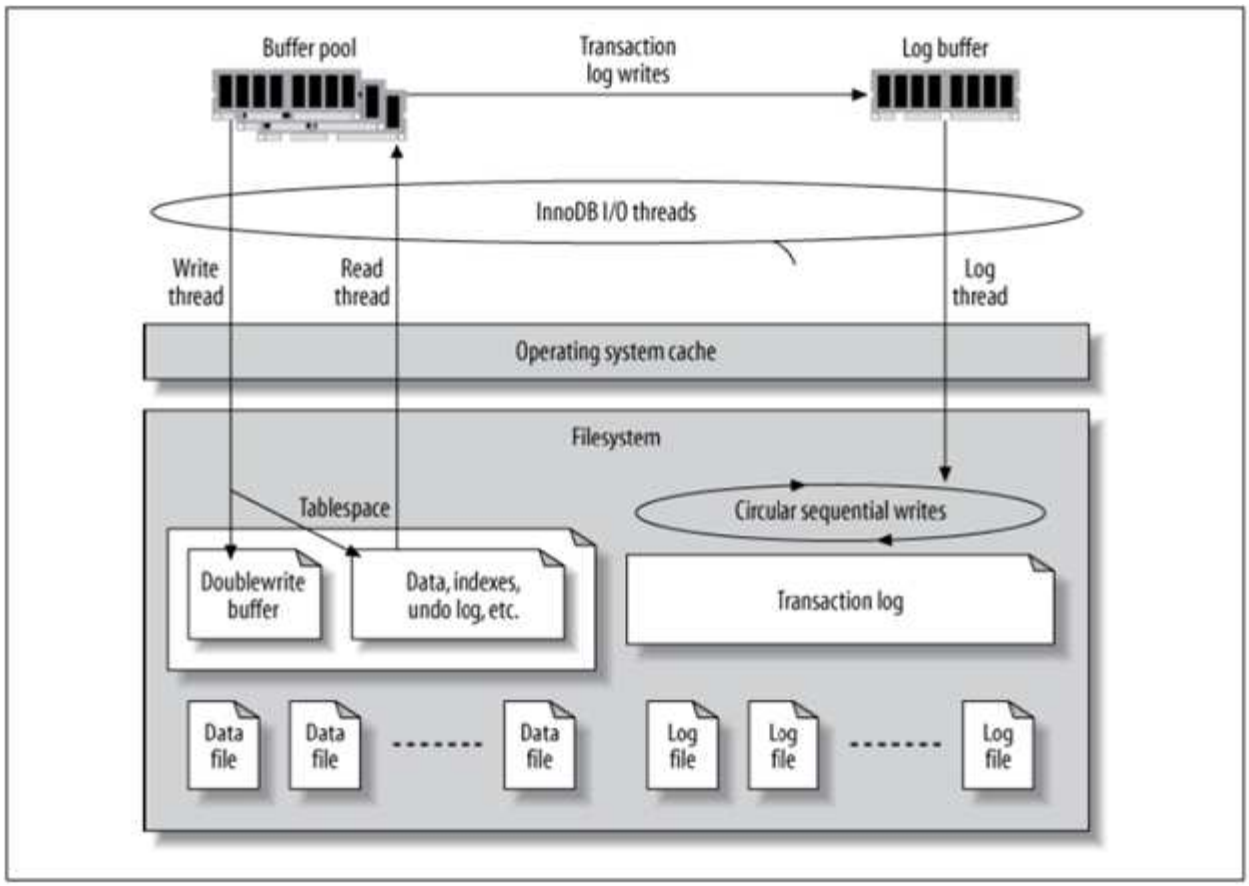
Zusätzliche Funktionen wie unterbrechungsfreie Upgrades (inkl. Storage-Austausch) stellen sicher, dass Ihre geschäftskritischen Datenbanken verfügbar bleiben. Darüber hinaus besteht die Möglichkeit zur sofortigen Disaster Recovery für große Umgebungen über MetroCluster oder zur Auswahl von Datenbanken mit SnapMirror Active Sync.

Am wichtigsten ist jedoch, dass ONTAP eine unübertroffene Performance sowie die Möglichkeit bietet, die Lösung entsprechend Ihren spezifischen Anforderungen zu dimensionieren. Unsere High-End-Systeme bieten über 1 Mio. IOPS bei Latenzen im Mikrosekundenbereich. Wenn Sie jedoch nur 100.000 IOPS benötigen, können Sie die Größe Ihrer Storage-Lösung mit einem kleineren Controller anpassen, auf dem dennoch genau dasselbe Storage-Betriebssystem ausgeführt wird.

Datenbankkonfiguration

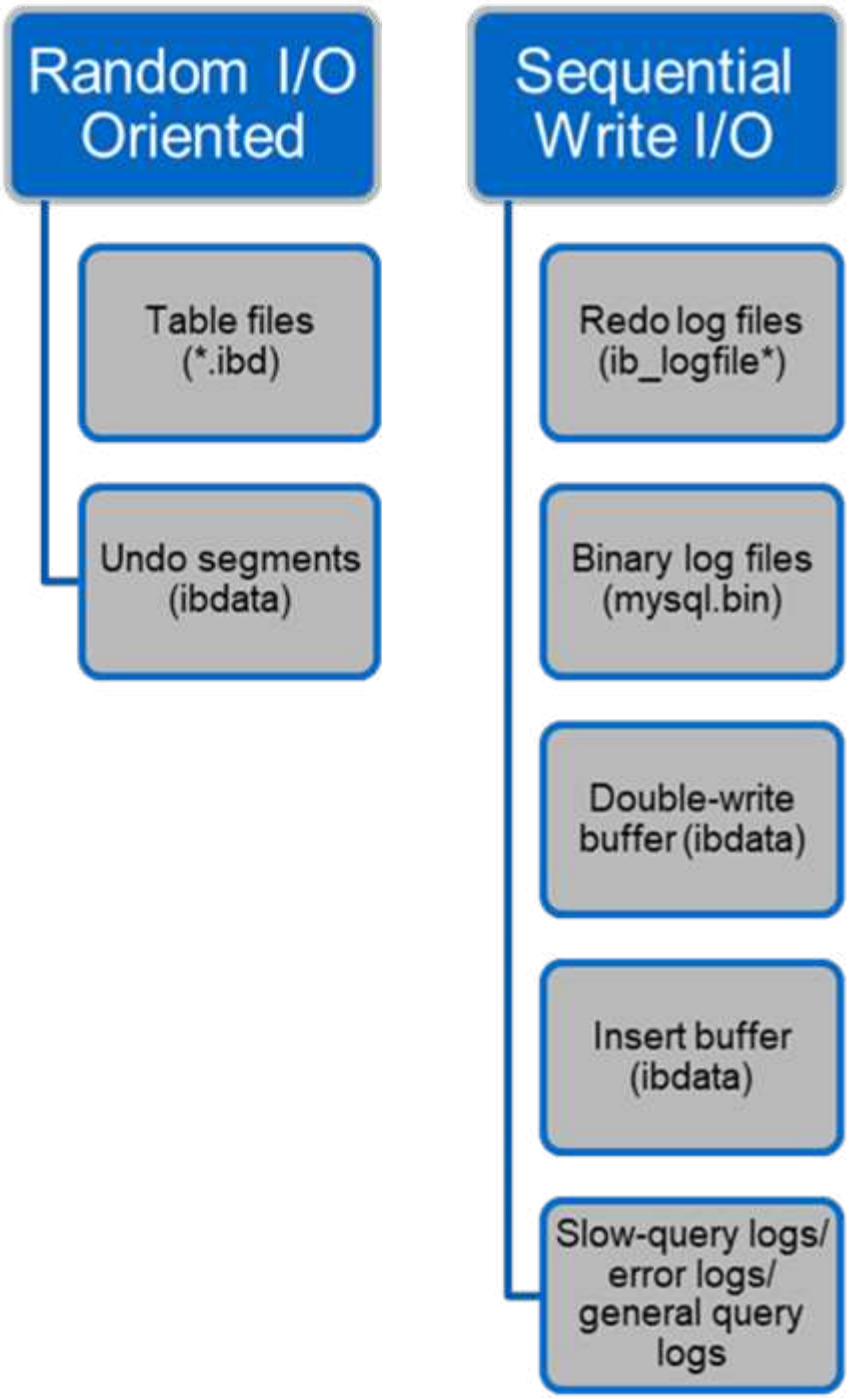
MySQL und InnoDB

InnoDB fungiert als mittlere Schicht zwischen Speicher und MySQL-Server, es speichert die Daten auf den Laufwerken.



MySQL I/O wird in zwei Typen unterteilt:

- Zufällige Datei-I/O
- Sequenzielle Datei-I/O



Datendateien werden zufällig gelesen und überschrieben, was zu einem hohen IOPS-Wert führt. Daher wird SSD-Speicher empfohlen.

Redo-Log-Dateien und binäre Log-Dateien sind Transaktionsprotokolle. Sie werden sequenziell geschrieben, sodass Sie mit dem Schreib-Cache eine gute Performance auf der Festplatte erzielen können. Ein sequenzieller Lesevorgang findet bei der Wiederherstellung statt, jedoch verursacht er selten ein Performance-Problem, da die Größe der Protokolldatei normalerweise kleiner ist als die von Datendateien und sequenzielle Lesevorgänge schneller sind als zufällige Lesevorgänge (die bei Datendateien auftreten).

Der Double-Write-Puffer ist eine Besonderheit von InnoDB. InnoDB schreibt zunächst gerötete Seiten in den Double-Write-Puffer und schreibt dann die Seiten an die richtigen Positionen in den Datendateien. Dieser

Prozess verhindert eine Beschädigung der Seite. Ohne den Double-Write-Puffer kann die Seite beschädigt werden, wenn während des Write-to-Drive-Prozesses ein Stromausfall auftritt. Da das Schreiben in den Doppelschreibpuffer sequenziell ist, wird es für HDDs stark optimiert. Bei der Wiederherstellung werden sequenzielle Lesevorgänge durchgeführt.

Da ONTAP NVRAM bereits einen Schreibschutz bietet, ist keine doppelte Schreibpufferung erforderlich. MySQL hat einen Parameter, `skip_innodb_doublewrite`, Um den Double-Write-Puffer zu deaktivieren. Diese Funktion kann die Leistung erheblich verbessern.

Der Insert-Puffer ist ebenfalls eine Besonderheit von InnoDB. Wenn sich nicht eindeutige sekundäre Indexblöcke nicht im Speicher befinden, fügt InnoDB Einträge in den Insert-Puffer ein, um zufällige I/O-Vorgänge zu vermeiden. Der Insert-Puffer wird in regelmäßigen Abständen in die sekundären Indexbäume der Datenbank eingebunden. Der Insert-Puffer reduziert die Anzahl der I/O-Vorgänge, indem I/O-Anfragen an denselben Block zusammengeführt werden. Zufällige I/O-Vorgänge können sequenziell sein. Der Einfügepuffer ist auch für HDDs stark optimiert. Sowohl sequenzielle Schreibvorgänge als auch Lesevorgänge erfolgen im normalen Betrieb.

Rückgängig-Segmente sind wahlfrei E/A-orientiert. Um die Multiversionsparallelität (MVCC) zu gewährleisten, muss InnoDB alte Bilder in den Undo-Segmenten registrieren. Beim Lesen vorheriger Bilder aus den Rückgängigmachungssegmenten sind zufällige Lesevorgänge erforderlich. Wenn Sie eine lange Transaktion mit wiederholbaren Lesevorgängen ausführen (wie z. B. mysqldump – einzelne Transaktion) oder eine lange Abfrage ausführen, können zufällige Lesevorgänge auftreten. Daher ist das Speichern von undo-Segmenten auf SSDs in dieser Instanz besser. Wenn Sie nur kurze Transaktionen oder Abfragen ausführen, stellen die zufälligen Lesevorgänge kein Problem dar.

NetApp empfiehlt aufgrund der InnoDB I/O-Eigenschaften das folgende Speicherdesign-Layout.

- Ein Volume zur Speicherung zufälliger und sequenzieller I/O-orientierter MySQL-Dateien
- Ein weiteres Volume zur Speicherung rein sequenzieller I/O-orientierter Dateien von MySQL

Dieses Layout hilft Ihnen auch bei der Entwicklung von Datensicherungsrichtlinien und -Strategien.

MySQL-Konfigurationsparameter

NetApp empfiehlt ein paar wichtige MySQL-Konfigurationsparameter, um eine optimale Performance zu erzielen.

Parameter	Werte
innodb_Log_file_size	256 MIO.
innodb_Flush_log_at_trx_commit	2
innodb_doublewrite	0
innodb_Flush_Method	Fsync
innodb_Buffer_Pool_size	11 G
innodb_io_Capacity	8192
innodb_Buffer_Pool_Instances	8
innodb_lru_Scan_depth	8192

Open_File_Limit	65535

Um die in diesem Abschnitt beschriebenen Parameter einzustellen, müssen Sie sie in der MySQL-Konfigurationsdatei (my.cnf) ändern. Die Best Practices von NetApp wurden in internen Tests durchgeführt.

innodb_Log_file_size

Die Auswahl der richtigen Größe für die InnoDB-Protokolldateigröße ist wichtig für die Schreibvorgänge und für eine anständige Wiederherstellungszeit nach einem Serverabsturz.

Da so viele Transaktionen in der Datei angemeldet sind, ist die Größe der Protokolldatei für Schreibvorgänge wichtig. Wenn Datensätze geändert werden, wird die Änderung nicht sofort in den Tablespace zurückgeschrieben. Stattdessen wird die Änderung am Ende der Protokolldatei aufgezeichnet und die Seite als verschmutzt markiert. InnoDB verwendet sein Protokoll, um zufällige I/O in sequenzielle I/O-Vorgänge zu konvertieren

Wenn das Protokoll voll ist, wird die fehlerhafte Seite nacheinander in den Tablespace geschrieben, um Speicherplatz in der Protokolldatei freizugeben. Angenommen, ein Server stürzt mitten in einer Transaktion ab, und die Schreibvorgänge werden nur in der Protokolldatei aufgezeichnet. Bevor der Server wieder live gehen kann, muss er eine Wiederherstellungsphase durchlaufen, in der die in der Protokolldatei aufgezeichneten Änderungen wiedergegeben werden. Je mehr Einträge in der Protokolldatei vorhanden sind, desto länger dauert es, bis der Server wiederhergestellt ist.

In diesem Beispiel wirkt sich die Größe der Protokolldatei sowohl auf die Wiederherstellungszeit als auch auf die Schreib-Performance aus. Wenn Sie die richtige Zahl für die Größe der Protokolldatei wählen, gleichen Sie die Recovery-Zeit mit der Schreib-Performance ab. Normalerweise ist alles zwischen 128M und 512M ein gutes Preis-Leistungs-Verhältnis.

innodb_Flush_log_at_trx_commit

Wenn eine Datenänderung vorgenommen wird, wird nicht sofort in den Storage geschrieben.

Stattdessen werden die Daten in einem Protokollpuffer aufgezeichnet, einem Teil des Speichers, den InnoDB Pufferänderungen zuweist, die in der Protokolldatei aufgezeichnet werden. InnoDB leert den Puffer in die Protokolldatei, wenn eine Transaktion durchgeführt wird, wenn der Puffer voll wird, oder einmal pro Sekunde, je nachdem, welches Ereignis zuerst eintritt. Die Konfigurationsvariable, die diesen Prozess steuert, ist innodb_flush_log_at_trx_commit. Die Wertoptionen umfassen:

- Wenn Sie es einstellen `innodb_flush_log_trx_at_commit=0`, InnoDB schreibt die geänderten Daten (im InnoDB-Pufferpool) in die Log-Datei (ib_logfile) und leert die Log-Datei (Write to Storage) jede Sekunde. Es tut jedoch nichts, wenn die Transaktion durchgeführt wird. Bei einem Stromausfall oder Systemabsturz können die nicht gespeicherten Daten nicht wiederhergestellt werden, da sie weder auf die Protokolldatei noch auf die Laufwerke geschrieben werden.
- Wenn Sie es einstellen `innodb_flush_log_trx_commit=1`, InnoDB schreibt den Protokollpuffer in das Transaktionsprotokoll und leert für jede Transaktion auf eine dauerhafte Speicherung. Beispielsweise schreibt InnoDB für alle Transaction Commits in das Protokoll und schreibt anschließend in den Speicher. Langsamer Speicher beeinträchtigt die Performance, beispielsweise wird die Anzahl der InnoDB-Transaktionen pro Sekunde reduziert.
- Wenn Sie es einstellen `innodb_flush_log_trx_commit=2`, InnoDB schreibt den Protokollpuffer bei

jedem Commit in die Protokolldatei, schreibt aber keine Daten in den Speicher. InnoDB leert die Daten einmal pro Sekunde. Selbst bei einem Stromausfall oder Systemabsturz sind die Daten von Option 2 in der Protokolldatei verfügbar und können wiederhergestellt werden.

Wenn die Leistung das Hauptziel ist, setzen Sie den Wert auf 2. Da InnoDB einmal pro Sekunde auf die Laufwerke schreibt, nicht für jede Transaktion, verbessert sich die Performance erheblich. Bei einem Stromausfall oder Absturz können die Daten aus dem Transaktions-Log wiederhergestellt werden.

Wenn die Datensicherheit das Hauptziel ist, setzen Sie den Wert auf 1, sodass InnoDB für jeden Transaktionscommit zu den Laufwerken leert. Möglicherweise ist die Performance jedoch beeinträchtigt.

NetApp empfiehlt Setzen Sie den innodb_flush_log_trx_commit Wert auf 2, um eine bessere Leistung zu erzielen.

innodb_doublewrite

Wenn `innodb_doublewrite` Ist aktiviert (Standard), speichert InnoDB alle Daten zweimal: Zuerst in den Double-Write-Puffer und dann in die eigentlichen Datendateien.

Sie können diesen Parameter mit deaktivieren `--skip-innodb_doublewrite` Für Benchmarks oder wenn Sie sich mehr um Top-Performance als um Datenintegrität oder mögliche Ausfälle sorgen. InnoDB verwendet eine Datei-Flush-Technik namens Double-Write. Bevor die Seiten in die Datendateien geschrieben werden, schreibt InnoDB sie in einen zusammenhängenden Bereich, den sogenannten Double-Write-Puffer. Nachdem das Schreiben und der Flush in den Double-Write-Puffer abgeschlossen sind, schreibt InnoDB die Seiten an die richtigen Positionen in der Datendatei. Falls das Betriebssystem oder ein mysqld-Prozess während eines Page Write abstürzt, kann InnoDB später während der Crash-Wiederherstellung eine gute Kopie der Seite aus dem Double-Write-Puffer finden.

NetApp empfiehlt den Double-Write-Puffer zu deaktivieren. ONTAP NVRAM dient dieselbe Funktion. Die doppelte Pufferung beeinträchtigt die Leistung unnötig.

innodb_Buffer_Pool_size

Der InnoDB Pufferpool ist der wichtigste Teil jeder Tuning-Aktivität.

InnoDB setzt stark auf den Pufferpool für das Caching von Indizes und Rudern der Daten, den adaptiven Hash-Index, den Insert-Puffer und viele andere interne Datenstrukturen. Der Puffer-Pool puffert Änderungen an Daten, damit Schreibzugriffe nicht sofort in den Storage übernommen werden müssen, was die Performance verbessert. Der Pufferpool ist integraler Bestandteil von InnoDB und muss entsprechend angepasst werden. Berücksichtigen Sie beim Festlegen der Größe des Puffer-Pools die folgenden Faktoren:

- Stellen Sie für eine dedizierte InnoDB-Maschine die Pufferpoolgröße auf 80 % oder mehr verfügbaren RAM ein.
- Wenn es sich nicht um einen dedizierten MySQL-Server handelt, stellen Sie die Größe auf 50 % des RAM ein.

innodb_Flush_Method

Der Parameter innodb_flush_method gibt an, wie InnoDB die Protokoll- und Datendateien öffnet und löscht.

Optimierungen

In der InnoDB-Optimierung wird durch die Einstellung dieses Parameters die Datenbankleistung ggf. optimiert.

Die folgenden Optionen sind für das Spülen der Dateien über InnoDB:

- `fsync`. InnoDB verwendet die `fsync()` Systemaufruf, um sowohl die Daten- als auch die Protokolldateien zu leeren. Diese Option ist die Standardeinstellung.
- `O_DSYNC`. InnoDB verwendet die `O_DSYNC` Option zum Öffnen und Leeren der Protokolldateien und fsync() zum Leeren der Datendateien. InnoDB wird nicht verwendet `O_DSYNC` Direkt, weil es Probleme mit ihm auf vielen Sorten von UNIX.
- `O_DIRECT`. InnoDB verwendet die `O_DIRECT` Option (oder `directio()` Unter Solaris), um die Datendateien zu öffnen und zu verwenden `fsync()` Um sowohl die Daten als auch die Protokolldateien zu löschen. Diese Option ist auf einigen GNU/Linux-Versionen, FreeBSD und Solaris verfügbar.
- `O_DIRECT_NO_FSYNC`. InnoDB verwendet die `O_DIRECT` Option beim Spülen von E/A, wird jedoch übersprungen `fsync()` Systemaufruf danach. Diese Option ist für einige Arten von Dateisystemen ungeeignet (z. B. XFS). Wenn Sie sich nicht sicher sind, ob Ihr Dateisystem einen erfordert `fsync()` Systemaufruf – zum Beispiel zum Beibehalten aller Dateimetadaten – verwendet den `O_DIRECT` Wählen Sie stattdessen.

Beobachtung

In den NetApp-Labortests ist der `fsync` Auf NFS und SAN kam die Standardoption zum Einsatz. Im Vergleich dazu war sie ein großartiger Performance-Improvisator `O_DIRECT`. Bei Verwendung der Spülmethode als `O_DIRECT` Bei ONTAP konnten wir beobachten, dass der Client viele Single-Byte-Schreibvorgänge am Rand des 4096. Blocks in serieller Form schreibt. Diese Schreibvorgänge haben die Latenz über das Netzwerk erhöht und die Performance beeinträchtigt.

innodb_io_Capacity

Im InnoDB Plug-in wurde ein neuer Parameter namens innodb_io_Capacity aus MySQL 5.7 hinzugefügt.

Er steuert die maximale Anzahl von IOPS, die InnoDB durchführt (einschließlich der Spülrate von schmutzigen Seiten sowie der Batch-Größe des Insert Buffer [ibuf]). Der innodb_io_capacity Parameter setzt eine Obergrenze für IOPS durch InnoDB-Hintergrundaufgaben, wie das Leeren von Seiten aus dem Pufferpool und das Zusammenführen von Daten aus dem Änderungspuffer.

Legen Sie den Parameter innodb_io_Capacity auf die ungefähre Anzahl von E/A-Vorgängen fest, die das System pro Sekunde durchführen kann. Halten Sie die Einstellung idealerweise so niedrig wie möglich, aber nicht so niedrig, dass Hintergrundaktivitäten langsamer werden. Ist die Einstellung zu hoch, werden die Daten aus dem Puffer-Pool entfernt und Puffer für das Caching zu schnell eingefügt, um einen wesentlichen Vorteil zu erzielen.

NetApp empfiehlt, wenn Sie diese Einstellung über NFS verwenden, das Testergebnis von IOPS (SysBench/FiO) analysieren und den Parameter entsprechend einstellen. Verwenden Sie den kleinstmöglichen Wert zum Spülen und Spülen, um mit dem Schritt zu bleiben, es sei denn, Sie sehen mehr geänderte oder schmutzige Seiten als Sie im InnoDB-Puffer-Pool möchten.

Verwenden Sie keine Extremwerte wie 20,000 oder mehr, es sei denn, Sie haben bewiesen, dass niedrigere Werte für Ihre Arbeitsbelastung nicht ausreichen.

Der InnoDB_IO_Capacity Parameter regelt Spülraten und zugehörige I/O.

> Sie können die Leistung ernsthaft beeinträchtigen, indem Sie diesen Parameter oder den innodb_io_Capacity_max-Parameter zu hoch einstellen und I/O-Operationen mit vorzeitigem Spülen verschwenden.

innodb_lru_Scan_depth

Der `innodb_lru_scan_depth` Parameter beeinflusst die Algorithmen und Heuristiken des Flush-Vorgangs für den InnoDB Pufferpool.

Dieser Parameter ist in erster Linie an Performance-Experten interessiert, die I/O-intensive Workloads optimieren. Dieser Parameter gibt für jede Pufferpoolinstanz an, wie weit der Seitenauflauf des Seitenreinigers in der LRU-Seitenliste (Least Recently Used) weiter scannen soll, und sucht nach verschmutzten Seiten, die bereinigt werden sollen. Dieser Hintergrundvorgang wird einmal pro Sekunde durchgeführt.

Sie können den Wert nach oben oder unten anpassen, um die Anzahl der freien Seiten zu minimieren. Stellen Sie den Wert nicht wesentlich höher ein als erforderlich, da die Scans erhebliche Performancekosten verursachen können. Ziehen Sie außerdem in Betracht, diesen Parameter anzupassen, wenn Sie die Anzahl der Pufferpool-Instanzen ändern, da `innodb_lru_scan_depth * innodb_buffer_pool_instances` Definiert den Umfang der Arbeit, die durch den Seitenreinigungsfaden jede Sekunde durchgeführt wird.

Eine Einstellung, die kleiner als die Standardeinstellung ist, eignet sich für die meisten Workloads. Ziehen Sie in Betracht, diesen Wert nur zu erhöhen, wenn Sie freie I/O-Kapazität bei einem typischen Workload haben. Wenn ein schreibintensiver Workload die I/O-Kapazität aussättigt, verringern Sie den Wert umgekehrt, insbesondere wenn ein großer Puffer-Pool vorhanden ist.

Open_File_Limits

Der `open_file_limits` Parameter bestimmt die Anzahl der Dateien, die das Betriebssystem mysqld zum Öffnen zulässt.

Der Wert dieses Parameters zur Laufzeit ist der vom System zulässige Realwert und kann sich von dem Wert unterscheiden, den Sie beim Serverstart angeben. Der Wert ist 0 auf Systemen, bei denen MySQL die Anzahl der geöffneten Dateien nicht ändern kann. Die effektive `open_files_limit` Der Wert basiert auf dem Wert, der beim Systemstart (falls vorhanden) angegeben wurde, und den Werten von `max_connections` Und `table_open_cache` Mit diesen Formeln:

- 10 + `max_connections` + (`table_open_cache` 2 x)
- `max_connections` X 5
- Betriebssystemgrenze, wenn positiv
- Wenn die Betriebssystemgrenze unendlich ist: `open_files_limit` Wert wird beim Start angegeben; 5,000 wenn keine

Der Server versucht, die Anzahl der Dateideskriptoren mit dem Maximum dieser vier Werte zu ermitteln. Wenn so viele Deskriptoren nicht abgerufen werden können, versucht der Server, so viele zu erhalten, wie das System zulässt.

Host-Konfiguration

MySQL-Containerisierung

Die Containerisierung von MySQL-Datenbanken nimmt immer mehr zu.

Das Low-Level-Container-Management wird fast immer über Docker durchgeführt. Container-Managementplattformen wie OpenShift und Kubernetes vereinfachen das Management großer Container-Umgebungen noch. Die Vorteile der Containerisierung sind niedrigere Kosten, da keine Hypervisor-Lizenz erforderlich ist. Darüber hinaus ermöglichen Container die Ausführung mehrerer Datenbanken isoliert voneinander, während gleichzeitig der gleiche zugrunde liegende Kernel und das gleiche Betriebssystem verwendet werden. Container können in Mikrosekunden bereitgestellt werden.

NetApp bietet mit Astra Trident erweiterte Management-Funktionen für Storage. Mit Astra Trident kann ein in Kubernetes erstellter Container automatisch seinen Storage auf der entsprechenden Tier bereitstellen, Richtlinien für den Export anwenden, Snapshot-Richtlinien festlegen und sogar einen Container auf einen anderen klonen. Weitere Informationen finden Sie im "Astra Trident-Dokumentation".

Slot-Tabellen für MySQL und NFSv3

Die NFSv3-Leistung unter Linux hängt von einem Parameter namens ab `tcp_max_slot_table_entries`.

TCP-Slot-Tabellen sind das NFSv3 Äquivalent zur Warteschlangentiefe des Host Bus Adapters (HBA). Diese Tabellen steuern die Anzahl der NFS-Vorgänge, die zu einem beliebigen Zeitpunkt ausstehen können. Der Standardwert ist normalerweise 16, was für eine optimale Performance viel zu niedrig ist. Das entgegengesetzte Problem tritt auf neueren Linux-Kerneln auf, die automatisch die Begrenzung der TCP-Slot-Tabelle auf ein Niveau erhöhen können, das den NFS-Server mit Anforderungen sättigt.

Um eine optimale Performance zu erzielen und Performance-Probleme zu vermeiden, passen Sie die Kernel-Parameter an, die die TCP-Slot-Tabellen steuern.

Führen Sie die aus `sysctl -a | grep tcp.*.slot_table` Und beobachten Sie die folgenden Parameter:

```
# sysctl -a | grep tcp.*.slot_table
sunrpc.tcp_max_slot_table_entries = 128
sunrpc.tcp_slot_table_entries = 128
```

Alle Linux-Systeme sollten enthalten `sunrpc.tcp_slot_table_entries`, Aber nur einige enthalten `sunrpc.tcp_max_slot_table_entries`. Beide sollten auf 128 gesetzt werden.

Achtung
Wenn diese Parameter nicht eingestellt werden, kann dies erhebliche Auswirkungen auf die Leistung haben. In einigen Fällen ist die Performance eingeschränkt, da das linux-Betriebssystem nicht genügend I/O ausgibt In anderen Fällen erhöht sich die I/O-Latenz, wenn das linux Betriebssystem versucht, mehr I/O-Vorgänge auszustellen, als gewartet werden kann.

I/O-Planer und MySQL

Der Linux-Kernel ermöglicht eine Steuerung auf niedriger Ebene über die Art und Weise, wie I/O-Vorgänge zum Blockieren von Geräten geplant werden.

Die Standardwerte auf verschiedenen Linux-Distributionen variieren erheblich. MySQL empfiehlt, dass Sie verwenden `NOOP` Oder A `deadline` I/O-Scheduler mit nativem asynchronem I/O (AIO) unter Linux. Im Allgemeinen zeigen NetApp Kunden und interne Tests mit NoOps bessere Ergebnisse.

Die InnoDB Storage Engine von MySQL verwendet das asynchrone I/O-Subsystem (native AIO) unter Linux, um Lese- und Schreibanforderungen für Datendateiseiten durchzuführen. Dieses Verhalten wird vom gesteuert `innodb_use_native_aio` Die standardmäßig aktivierte Konfigurationsoption. Bei nativem AIO hat der Typ des I/O-Planers einen größeren Einfluss auf die I/O-Performance. Durchführung von Benchmarks zur Bestimmung des I/O-Planers, der die besten Ergebnisse für Ihren Workload und Ihre Umgebung liefert

Anweisungen zur Konfiguration des I/O-Planers finden Sie in der entsprechenden Dokumentation zu Linux und MySQL.

MySQL-Dateideskriptoren

Zum Ausführen benötigt der MySQL-Server Dateideskriptoren, und die Standardwerte reichen nicht aus.

Sie werden verwendet, um neue Verbindungen zu öffnen, Tabellen im Cache zu speichern, temporäre Tabellen zum Beheben komplizierter Abfragen zu erstellen und auf persistente zuzugreifen. Wenn mysqld nicht in der Lage ist, neue Dateien zu öffnen, wenn nötig, kann es nicht mehr richtig funktionieren. Ein häufiges Symptom dieses Problems ist Fehler 24, „zu viele geöffnete Dateien“. Die Anzahl der Dateideskriptoren, die mysqld gleichzeitig öffnen kann, wird durch das definiert `open_files_limit` In der Konfigurationsdatei festgelegte Option (`/etc/my.cnf`). Aber `open_files_limit` Hängt auch von den Grenzen des Betriebssystems ab. Diese Abhängigkeit macht die Einstellung der Variable komplizierter.

MySQL kann seine Einstellung nicht festlegen `open_files_limit` Option höher als unter angegeben `ulimit 'open files'`. Daher müssen Sie diese Grenzwerte explizit auf Betriebssystemebene festlegen, damit MySQL Dateien nach Bedarf öffnen kann. Es gibt zwei Möglichkeiten, die Dateibegrenzung in Linux zu überprüfen:

- Der `ulimit` Befehl schnell gibt Ihnen eine detaillierte Beschreibung der Parameter, die erlaubt oder gesperrt werden. Die durch die Ausführung dieses Befehls vorgenommenen Änderungen sind nicht dauerhaft und werden nach einem Neustart des Systems gelöscht.
- Änderungen an `/etc/security/limit.conf` Die Datei ist dauerhaft und wird durch einen Systemneustart nicht beeinträchtigt.

Stellen Sie sicher, dass Sie sowohl die harten als auch die weichen Grenzwerte für Benutzer mysql ändern. Die folgenden Auszüge stammen aus der Konfiguration:

```
mysql hard nofile 65535
mysql soft nofile 65353
```

Aktualisieren Sie gleichzeitig dieselbe Konfiguration in `my.cnf` Um die offenen Dateigrenzen vollständig zu verwenden.

Storage-Konfiguration

MySQL mit NFS

Die MySQL-Dokumentation empfiehlt, NFSv4 für NAS-Bereitstellungen zu verwenden.

ONTAP NFS-Übertragungsgrößen

Standardmäßig beschränkt ONTAP die NFS-I/O-Größe auf 64K. Zufällige IO mit einer MySQL-Datenbank verwendet eine viel kleinere Blockgröße, die weit unter dem 64K Maximum liegt. I/O mit großen Blöcken wird in der Regel parallelisiert, sodass die 64K-Maximalgröße ebenfalls keine Einschränkung darstellt.

Es gibt einige Workloads, bei denen das 64K-Maximum eine Einschränkung darstellt. Insbesondere Vorgänge mit einem Thread, wie z. B. Backup-Vorgänge mit vollständiger Tabelle, werden schneller und effizienter ausgeführt, wenn die Datenbank weniger, aber größere I/O-Vorgänge ausführen kann. Die optimale I/O-Handhabungsgröße für ONTAP mit Datenbank-Workloads beträgt 256.000. Die unten aufgeführten NFS-Mount-Optionen für spezifische Betriebssysteme wurden entsprechend von 64K auf 256K aktualisiert.

Die maximale Übertragungsgröße für eine bestimmte ONTAP SVM kann wie folgt geändert werden:

```
Cluster01::> set advanced

Warning: These advanced commands are potentially dangerous; use them only
when directed to do so by NetApp personnel.

Do you want to continue? {y|n}: y

Cluster01::*> nfs server modify -vserver vserver1 -tcp-max-xfer-size
262144
```

Verringern Sie niemals die maximal zulässige Übertragungsgröße auf ONTAP unter den Wert rsize/wsize der aktuell gemounteten NFS-Dateisysteme. Dies kann bei einigen Betriebssystemen zu Hängebleiben oder sogar Datenbeschädigungen führen. Wenn beispielsweise NFS-Clients derzeit auf 65536 rsize/wsize gesetzt sind, dann könnte die maximale Übertragungsgröße für ONTAP ohne Auswirkung auf die Clients selbst begrenzt werden, zwischen 65536 und 1048576 angepasst werden. Wenn Sie die maximale Übertragungsgröße unter 65536 verringern, können die Verfügbarkeit oder die Daten beeinträchtigt werden.

NetApp empfiehlt

Einstellen der folgenden Einstellung für NFSv4 fstab (/etc/fstab):

```
nfs4 rw,
hard,nointr,bg,vers=4,proto=tcp,noatime,rsize=262144,wsize=262144
```

Ein häufiges Problem mit NFSv3 waren die gesperrten InnoDB-Protokolldateien nach einem Stromausfall. Durch die Verwendung von Zeit oder das Wechseln von Protokolldateien wurde dieses Problem behoben. NFSv4 verfügt jedoch über Sperrvorgänge und verfolgt die offenen Dateien und Delegationen.

MySQL mit SAN

Es gibt zwei Optionen zur Konfiguration von MySQL mit SAN unter Verwendung des üblichen zwei-Volume-Modells.

Kleinere Datenbanken können auf einem Standard-LUN-Paar platziert werden, sofern die I/O- und Kapazitätsanforderungen innerhalb der Grenzen eines einzigen LUN-Filesystems liegen. Eine Datenbank, die etwa 2.000 zufällige IOPS benötigt, kann beispielsweise auf einem einzelnen Filesystem auf einer einzelnen LUN gehostet werden. In ähnlicher Weise würde eine Datenbank mit einer Größe von nur 100 GB auf eine einzige LUN passen, ohne ein Management-Problem zu verursachen.

Bei größeren Datenbanken sind mehrere LUNs erforderlich. Beispielsweise würde eine Datenbank, die 100.000 IOPS benötigt, höchstwahrscheinlich mindestens acht LUNs benötigen. Eine einzelne LUN würde zu einem Engpass werden, da die Anzahl der SCSI-Kanäle zu Laufwerken nicht ausreicht. Eine 10-TB-Datenbank wäre ähnlich schwierig zu managen auf einer einzigen 10-TB-LUN. Logical Volume Manager sind so konzipiert, die Performance- und Kapazitätsfunktionen mehrerer LUNs miteinander zu verbinden, um Performance und Management zu verbessern.

In beiden Fällen sollte ein Paar ONTAP Volumes ausreichend sein. Bei einer einfachen Konfiguration würde die Datendatei-LUN wie die Protokoll-LUN in ein dediziertes Volume platziert werden. Bei einer logischen Volume Manager-Konfiguration befinden sich alle LUNs in der Datendatei-Volume-Gruppe in einem dedizierten Volume, und die LUNs der Log-Volume-Gruppe befinden sich in einem zweiten dedizierten Volume.

NetApp empfiehlt, zwei Dateisysteme für MySQL-Bereitstellungen auf SAN zu verwenden:

- Das erste Dateisystem speichert alle MySQL-Daten einschließlich Tablespace, Daten und Index.
- Das zweite Filesystem speichert alle Protokolle (binäre Protokolle, langsame Protokolle und Transaktionsprotokolle).

Es gibt mehrere Gründe für die Trennung von Daten auf diese Weise, unter anderem:

- Die I/O-Muster von Datendateien und Protokolldateien unterscheiden sich. Eine Trennung würde mehr Optionen mit QoS-Steuerung ermöglichen.
- Für eine optimale Nutzung der Snapshot Technologie müssen Datendateien unabhängig wiederhergestellt werden können. Das Kommingleren von Datendateien mit Protokolldateien beeinträchtigt die Wiederherstellung von Datendateien.
- NetApp SnapMirror bietet zwar eine einfache Disaster Recovery-Funktion mit einem geringen RPO für eine Datenbank, erfordert jedoch unterschiedliche Replizierungspläne für die Dateien und Protokolle.

Mit diesem grundlegenden Layout für zwei Volumes wird die Lösung zukunftssicher, sodass alle ONTAP Funktionen bei Bedarf genutzt werden können.

NetApp empfiehlt das Formatieren Ihres Laufwerks mit dem ext4-Dateisystem aufgrund der folgenden Features:

- Erweiterter Ansatz für Blockverwaltungsfunktionen, die im Journaling-Dateisystem (JFS) verwendet werden, und verzögerte Zuweisungsfunktionen des erweiterten Dateisystems (XFS).
- Ext4 erlaubt Dateisysteme mit bis zu 1 Exbibyte (2^60 Bytes) und Dateien mit bis zu 16 Tebibyte (16 * 2^40 Bytes). Im Gegensatz dazu unterstützt das ext3-Dateisystem nur eine maximale Filesystem-Größe von 16 TB und eine maximale Dateigröße von 2 TB.
- In ext4-Dateisystemen weist die Multiblockzuweisung (mballoc) mehreren Blöcken einer Datei in einem einzigen Vorgang zu, anstatt sie einzeln zuzuweisen, wie in ext3. Diese Konfiguration reduziert den Overhead beim mehrmaligen Aufrufen des Block Allocator und optimiert die Zuweisung von Speicher.
- Obwohl XFS der Standard für viele Linux-Distributionen ist, verwaltet es Metadaten anders und ist nicht für einige MySQL-Konfigurationen geeignet.

NetApp empfiehlt die Verwendung von 4k-Blockgrößenoptionen mit dem mkfs-Dienstprogramm, um die bestehende Block-LUN-Größe auszurichten.

```
mkfs.ext4 -b 4096
```

NetApp LUNs speichern Daten in physischen 4-KB-Blöcken, wodurch acht logische 512-Byte-Blöcke entstehen.

Wenn Sie nicht dieselbe Blockgröße einrichten, werden I/O-Vorgänge nicht korrekt an physischen Blöcken ausgerichtet und können in zwei verschiedene Laufwerke in einer RAID-Gruppe geschrieben werden, was zu Latenz führt.

Es ist wichtig, dass Sie den I/O so ausrichten, dass reibungslose Lese-/Schreibvorgänge erfolgen. Wenn die I/O jedoch bei einem logischen Block beginnt, der nicht am Anfang eines physischen Blocks liegt, ist die I/O falsch ausgerichtet. I/O-Vorgänge werden nur ausgerichtet, wenn sie an einem logischen Block beginnen – dem ersten logischen Block in einem physischen Block.